RTL COMPILER

Product Design is more than just Engineering

MANOJ.B

ACKNOWLEDGEMENT

I am thankful to my guide and instructor, Dr. J.V.R.Ravinrda,IIIT , Hyderabad for giving me this wonderful opportunity to write this book under his guidance. The love and faith that he showered on me have helped me to go on and complete this manual.

In the end, we would like to thank my friends, my parents and that invisible force that provided moral support and spiritual strength, which helped us completing this project successfully.

Table of contents **Page no.**

1.1 Introduction

The basic aim of an Encounter RTL compiler is to get a physical design from the netlist given to it by over verilog code. The physical design is acquired on a platform called GRAPHICAL USER INTERFACE (GUI) . as the name says it all, it is a interface between the user and the graphic design.

Encounter RTL complier follows Top-down global RTL design synthesis , that means the behavioural logic that we have written , will give you an overview of the system is formulated, specifying but not detailing any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements.

Due to top- down approach silicon realization will get done very rapidly in terms of speed and simplicity , because in a silicon realisation process , the area required for placement and routing concepts must be introduced primarily and that requires lot of information about the system module than the indivual modules.

This is a sample Physical design of a traffic lights program on how our netlist gets synthesized .

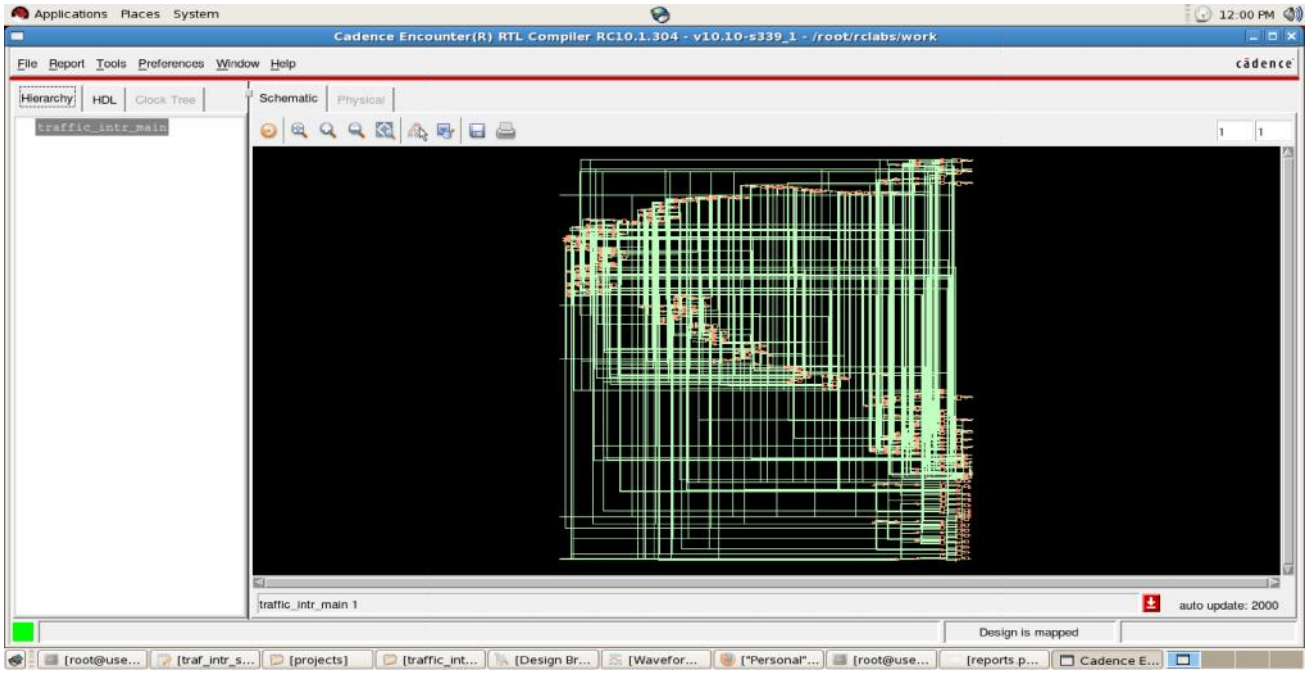


Fig I.1 A typical Physical Design (PD)

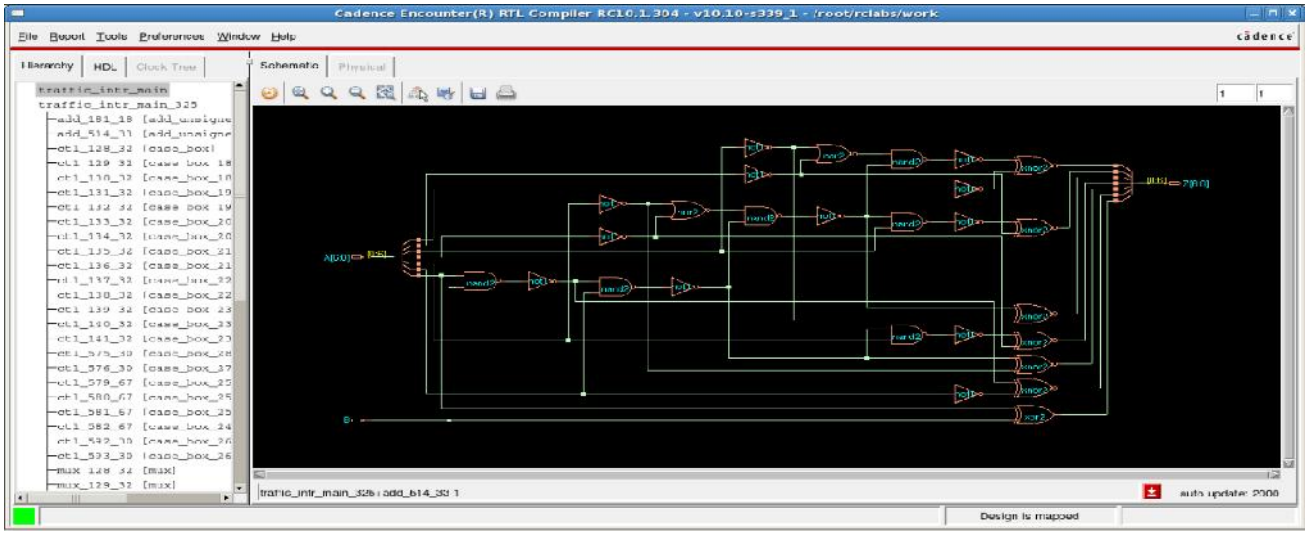


Fig I.2. This is an individual module inside the whole traffic control system

TIMING, AREA, AND POWER INTENT (PAT) IMPORTANCE

According to MOORE ,"after every 18 months duration the no. of transistors used on a chip gets doubled, the power consumed by that circuit gets decreased by a half, the size of the circuit is decreased to half and the frequency of the circuit gets doubled."

This rule is traditionally coming true due to the increasing thirst or need of man for the upgrading his comforts in his way of living , so the circuits has to take care of the three main factors called

- ✓ Power
- ✓ Timing
- ✓ Transistors

This thumb rule is followed by every VLSI company to sustain in the market , so we are designing the most optimised design for reducing the complexity or the congestion , which are at two different peaks , because if you decrease the complexity , the congestion gets decreased and when the congestion is getting increased, the complexity gets reduced, Design verification is all about balancing these two factors.

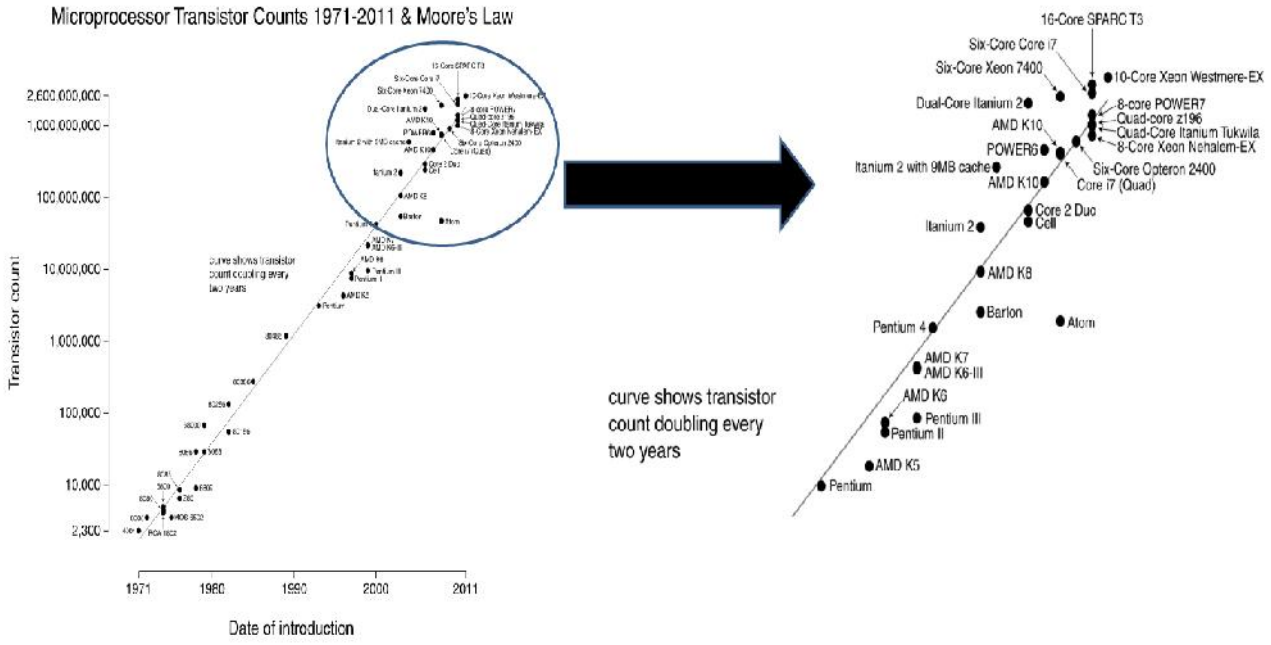


Fig I.3 Moore's law's Plot of CPU transistor counts against dates of introduction.

Chapter 1

BASIC DEFINITIONS

Encounter RTL Compiler consists of many more typical terms that is much different from outside world and the basic terms are as follows :-

1. **NETLIST** :- structural code

2. **layout and schematic :-** a schematic is the electrical hook up or connections of the circuit, but a layout is the physical representation of the PCB with actual shape of the parts i.e. connection to the PCB.

3. **Graphic User Interface (GUI):-** it is a output file of a RTL compiler, or the schematic design of our verilog code. It is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation

4. **SYNTHESIS** :-"Timing Check"
    1. Process of converting and mapping technology independent HDL into Technology dependent netlist

   it follows three steps ---> TOM

    b. **Translation** :- HDL to logical or Boolean operations
    c. **Optimisation :-** selecting the best out of the alternatives , i.e. designing with most minimized errors.
    d. **Mapping** :- technology independent to technology dependent

5. **simulation** :- "Functionality Check"
   its basic synthesis is to meet the target frequencies

6. **DFT** :- "Design for Testability" is a design techniques that add certain testability features to a
    i. microelectronic hardware product design
        - it can use few non- synthesizable codes also
7. **constraints** :- decides the perfect situation components but when given the right specifications

- EG :- just like a girl has restrictions from her father.

8. **Linting tool** :- it is a advance verification tool. It shows the error in the module (blocks ,paths, FSM,code) displays on the report file.

9. **wire load model** (WLM) :- Wire load modeling allows us to estimate the effect of wire length and fanout on the resistance, capacitance, and area of nets. Synthesizer uses these physical values to calculate wire delays and circuit speeds. Semiconductor vendors develop wire load models, based on statistical information specific to the vendors' process. The models include coefficients for area, capacitance, and resistance per unit length, and a fanout-to-length table for estimating net lengths (the number of fanouts determines a nominal length).

10. **non linear delay model** (NLDM) :- standard delay-modelling format.

11. **clock gating** :- this occurs when data signal is used to control the data . Output of a FF is dependent on input data.

12. **Library** :- A library is a collection of control attributes, environment description , standard cell description , delay calculations and delay models.

13. **LEF** :- Library Exchange Format :- contains specifications on layer, design rules, via definitions, metal capacitance ,cell descriptions, cell dimensions, layout of pins and blockages, capacitances in ASCII format.

14. **synthesis** :- refers to Logic synthesis, the process of converting a higher-level form of a design into a lower-level implementation

15. **DEF** :-**Design Exchange Format** is an open specification for representing physical layout of an integrated circuit in an ASCII format. It represents the net list and circuit layout. DEF is used in conjunction with Library Exchange Format (LEF) to represent complete physical layout of an integrated circuit while it is being designed.

16. **latch** :- Latches are designed to be *transparent.* That is, input signal changes cause immediate changes in output; when several *transparent* latches follow each other, using the same clock signal, signals can propagate through all of them at once.

    In electronics, a **flip-flop** or **latch** is a circuit that has two stable states and can be used to store state information. A flip-flop is a bistable multivibrator. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. It is the basic storage element in sequential logic. Flip-flops and latches are a fundamental building block of digital electronics systems used in computers, communications, and many other types of systems

SCAN D FLIPFLOP

A special type of D flip flop where it is functioned only when the test pattern is loaded . The idea is to drive the DFF 's D unit with an alternate source of data during device testing. After one or more clock ticks, the flipflops are put back into test mode, and the test results are "scanned out".

17. **FPGAs** are semi custom devices generally maintained for multiple times programming a single circuit, whereas ASICs are fully custom designs the circuit can no more be programmed when it is working under ASIC.

there is a weird element called one time programmable FPGA where you can actually program only for one purpose it is a SRAM, which has volatile memory.

18. **Mask-Programmable Gate Array (MPGA)** was developed to handle larger logic circuits. MPGAs consist of an array of pre-fabricated transistors that can be customized into the user logic circuit by connecting the transistors with custom wires.
-large setup cost is involved and manufacturing time is long.

FPGA vs. ASIC Design Advantages

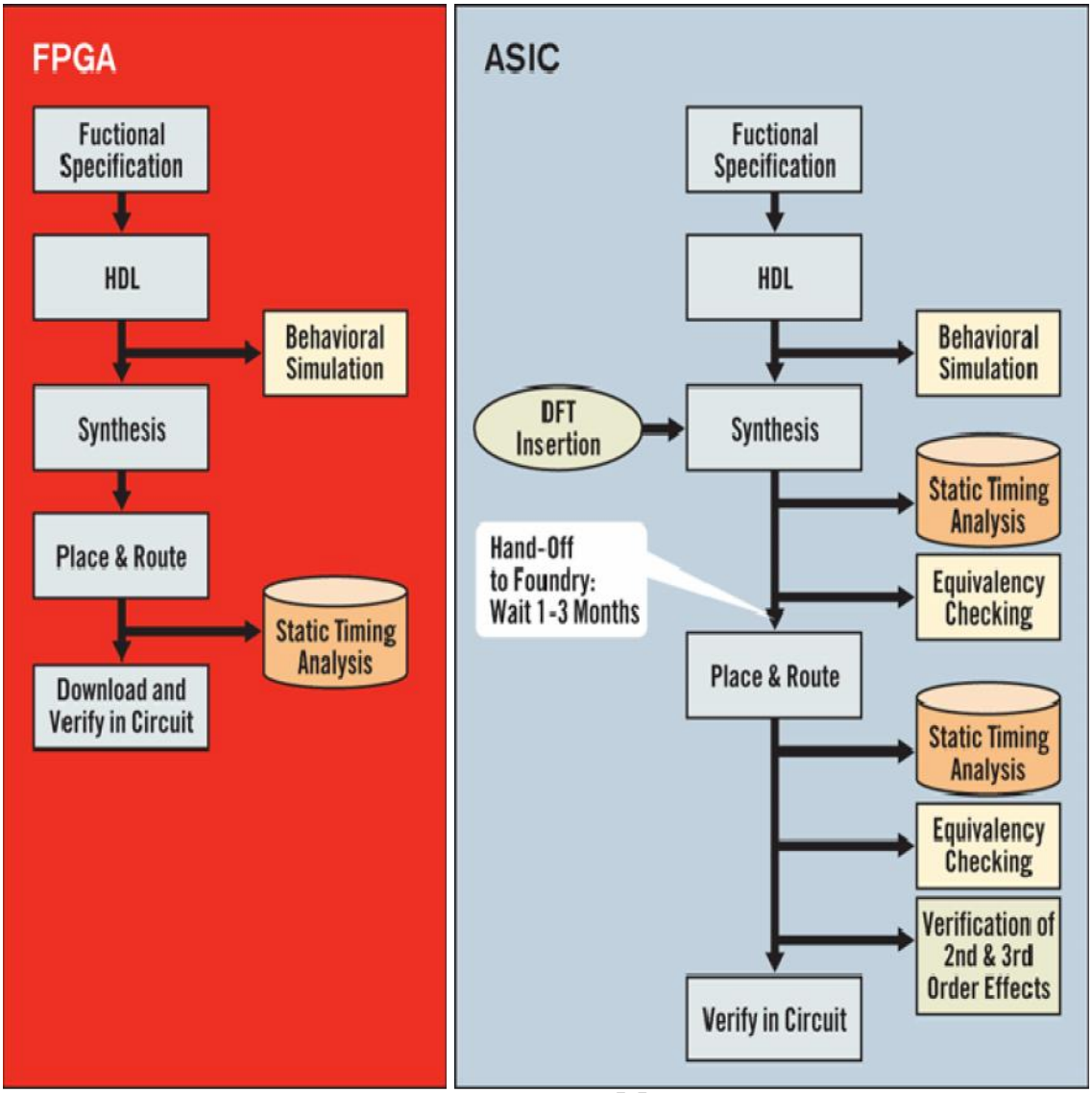
FPGA Design

Advantage	Benefit
Faster time-to-market	No layout, masks or other manufacturing steps are needed
No upfront non-recurring expenses (NRE)	Costs typically associated with an ASIC design
Simpler design cycle	Due to software that handles much of the routing, placement, and timing
More predictable project cycle	Due to elimination of potential re-spins, wafer capacities, etc.
Field reprogram ability	A new bit stream can be uploaded remotely

ASIC Design

Advantage	Benefit
Full custom capability	For design since device is manufactured to design specs
Lower unit costs	For very high volume designs
Smaller form factor	Since device is manufactured to design specs

FPGAs used for lower speed/complexity/volume designs in the past, today’s FPGAs easily push the 500 MHz performance barrier. the examples are embedded processors,DSP blocks, clocking, high-speed serial at ever lower price point



ASIC and FPGA modelling

19. **JTAG** (Joint test action group) it is a process for the boundary scan for printed circuit boards. It can be implemented on a processor only if it has enough pins on it.

20. **BOTTOM - UP APPROACH** :- combination of small modules each one after another after synthesizing and optimisation.

21. **TOP - DOWN APPROACH** :-synthesizing and optimising directly the big module which is a combination of small modules.

22. **Path groups**

Typically Path groups are the connections in between the modules is divided into four
They are

- input to output path group
- input to register path group
- register to register path group
- register to output path group

all these path groups are essentially should be out of congestion

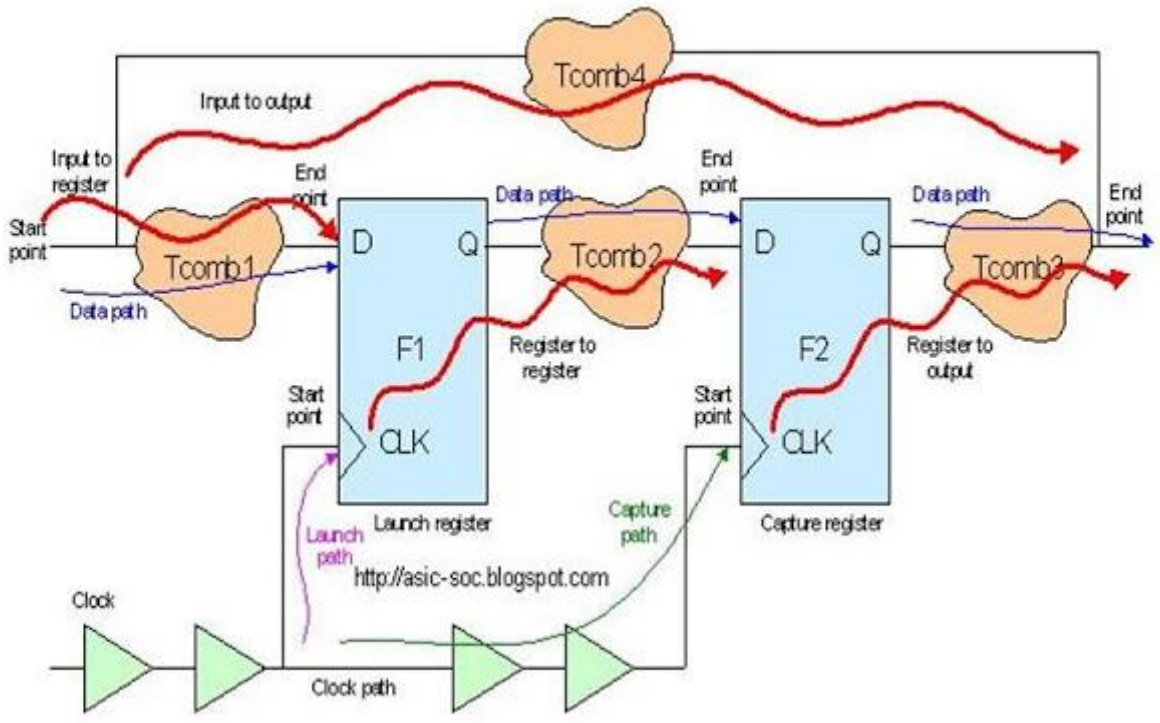


Fig 1.3 Path groups

Chapter 2

Compilation Procedure

For making use of our raw verilog code there is a need to generate the netlists , for generation of these netlists we need to synthesize our own code first. Inputs of this synthesis are our verilog code(RTL) , library files , our constraints and the reports and netlists will be our outputs.

library files + RTL code + constraints + (LEF & DEF files ,optional) = gate netlist(optimised) + scripts/reports

is the basic equation of SYNTHESIS, it is nothing but just translation of HDL to netlists. The output includes the First Encounter input files.

Synthesis is a technology independent process. The basic design flow is followed in this way

1. **HDL coding**
   The basic idea you get for designing a circuit, its logic, can be programmed on a platform called verilog in
   a. dataflow level
   b. structural level
   c. behavioural level

   Writing structural verilog code is a difficult task when the circuit complexity increases. Behavioural code of verilog is the simple way of writing a program of a logic. Machine can convert it into gate level netlists.

2. **set paths & libraries**

   the tool needs to load the library files (slow, fast, nominal libraries) to use the verilog logic given by the programmer , generally libraries contains all the basic gates with different fan-ins, fan-outs, and also with different number of inputs (e.g.:- 4 input NAND), flops, level shifters, muxes, etc .

A set of values of PVT(Process, Voltage, Temperature) is known as **operating condition**. A logic library is characterized for one set of operating condition. Generally there are different libraries specific to different operating condition. There are three operating conditions very commonly used in ASIC synthesis and implementation. Based on the affect on cell delay due to the variation in PVT these classifications are made.

They are:

- ✓ **worst** (also called ‘max’ or ‘slow’) library in which cells are characterized for maximum delay
- ✓ **best**(also called ‘min’ or ‘fast’)library in which cells are characterized for minimum delay
- ✓ **nominal**(also called ‘typical’ or ‘normal’)library in which cells are characterized for typical delay

3.reading the HDL files

reading the netlist (hardware description language) must be done as soon as the synthesis is done , we specify the netlist file as *.vg .

4 Elaborate designs

Graphic User Interface is a design which gives you the complete layout of this circuit . so a seperate command called **elaborate** is given in the TCL file for this layout generation .

3. Specify constraints

for optimising a specific design in the desired congestion, area , power and timing we use various constraint files to limit the layout only to specific conditions.

4. **synthesize**

   synthesis technically mean breaking up the things , similarly in VLSI world synthesize mean to break the logic of the program into electronic devices.

5. **check results**

   checking the result such as timing slack, congestion ,power, area after the compilation is to be checked after every optimisation step is must , that will decide the endpoint of optimisation. If not met the optimised point feedback the process, add few more constraints to it.

6. **netlists and constraints**

   Now generate the netlist using the most favourable constraints that gives you the specific requirements.

Just like a barren lands which do not have much minerals, a BAD NETLIST has poorly structured paths which will give you higher cell area. Bad netlist not only gives you high cell area but also it is too slow, has more iterations and is unroutable in many cases. World loves easy circuits, i.e. which has easy layout and easy timing but not congestion routing
where even buffers are used .
note:- buffers are used for getting a time delay but introduction of clocks increases power consumption as these instances also consume some amount of power.

Generally ASIC designs are made to synthesize with the help of RTL compiler. as a writer I must make sure that my reader understands clearly what is an ASIC and a FPGA. FPGA cell is a vast, huge ,big thing i.e it contains many ASIC designs of different functionalities but made to work on same operation .

The standard factors of optimising a design are

1. complexity
2.congestion (cannot be optimised much during RTL compilation, Soc Encounter is best for checking and optimising this constraint)
3.timing (slack)
4.power
5.area

the conventional ways of optimising a function mathematically is by

1. k-mapping
2.sum of products(SOP)
3.product of sums(POS)

similarly in optimising a physical design in RTL compiler world we use several parameters such as
1.generic structuring
2.global structuring
3.incremental synthesis

but we use them inside the terminal command language (tcl) file during RTL compiling stage .
Apart from optimising the complexity of the function we can even increase/decrease run time for optimising the timing constraints by making the effort levels as
1.low
2.medium
3.high

the three different levels experience different levels of runtime. for example runtime of low is x , then run time of medium is x^2 and runtime of high is x^3

Chapter 3

Flow of RTL compiler

Go and check the finest verilog file before computing

note:- 1.It must not include any "assign" statements in continuous blocks .
2.Place all your assign statements in procedural block

WHEN IT COMES TO COMPLEX DESIGN , DTMF (Dual Tone Multi-Frequency model)

0. Check out the verilog file
1. Go to set up file and copy that into a new tcl file.
2. Give the necessary file conditions inside the setup file.
3. setup.tcl file includes all the necessary conditons needed for the module level synthesis over the DTMF VERILOG CODE
4. generate a template file in the present working directory
5. give the necessary conditions to it

- # followed by any command is commented out in .tcl files.

- Index 1 is a set of values taken as 1 variable under considerations where as index 2 is a set of values taken as another variable under considerations. "Fast.lib" contains only level shifter cells

BASIC COMMANDS TO BE USED

Compilation

1.**rc -gui** :- dives you graphical interface of your verilog code, in short we wil get the schematic of our code.

2.**rc -f** :- the log file gets dispayed during the execution, as well as the power , area ,timing also gets displayed in this log file.

3.**rc -gui -f setup.tcl** :- execute your setup.tcl file and give this command

4.**rc -version** :- gives you the present version information Program Name: Encounter(R) RTL Compiler, Version: RC10.1.304 - v10.10-s339_1 (32-bit)

5.**rc -f run_syn.tcl -logfile mj.tcl** :- gives you the log file in mj.tcl editor that contain evrything that displayed during the time of execution.

6.**help command_name or man command_name**:-for help on all commands

7. **write_template -simple -outfile template.tcl**

generates a .tcl file named template

Inside **the program**

setting an attribute

1.**set_attribute <attribute name> <value> <objects> :**

set_attribute lib_search_path/root/rclabs/libraries/TIMING/STDCELL :-

from this command the library files are taken the list of files available inside STDCELL are

```
-rw-r--r-- 1              fast.lib
-rw-r--r-- 1            ff_g_1v32_0c.lib
-rw-r--r-- 1            ff_hvt_1v32_0c.lib
-rw-r--r-- 1               README
-rw-r--r-- 1             slow_0v70_1v08.lib
-rw-r--r-- 1             slow_0v80_0v80.lib
-rw-r--r-- 1             slow_0v80_0v90.lib
-rw-r--r-- 1             slow_0v80_1v08.lib
-rw-r--r-- 1             slow_0v90_1v08.lib
-rw-r--r-- 1             slow_1v08_1v08.lib
-rwxr-xr-x 1            slow.lib
-rw-r--r-- 1            ss_g_0v70_125c.lib
-rw-r--r-- 1            ss_g_0v80_125c.lib
-rw-r--r-- 1            ss_g_0v90_125c.lib
-rw-r--r-- 1            ss_g_1v08_125c.lib
-rw-r--r-- 1            ss_hvt_0v70_125c.lib
-rw-r--r-- 1            ss_hvt_0v80_125c.lib
-rw-r--r-- 1            ss_hvt_0v90_125c.lib
-rw-r--r-- 1            ss_hvt_1v08_125c.lib
-rw-r--r-- 1              ss_pg_0v70_1v08.lib
-rw-r--r-- 1               ss_pg_0v70_1v08.lib_org
-rw-r--r-- 1            tt_g_1v20_25c.lib
-rw-r--r-- 1            tt_hvt_1v20_25c.lib
-rw-r--r-- 1              typical.lib
```

2.**set_attribute hdl_search_path /root/rclabs/work**

giving you the HDL file search path

3.**get_attribute <attribute_name> <object>**

retrieving of attributes ,
volatile in nature, prime difference bwtween set and get is that we can change the attribute over get_attr that means it follows multiple paths but in set_attr program synthesis is relying only on only path .

4. **set_attr library lsi50k.lib /**

to load a single library file

5. **set_attr library {{a.lib b.lib} c.lib {x.lib y.lib}} /**

RTL comipiler loads a.lib and appends b.lib to a.lib. then it loads c.lib. next it loads x.lib and appends y.lib to x.lib.

6. **set_attr avoid <true(1)/false(0)> <cell name(s)>**
set_attr avoid 1 {mylib/sn1_mux21_prx*}

can avoid the sn1_mux21_prx* library during synthesis

7.**read_hdl -v2001 counter.v**

reads your verilog code

read_netlist -v2001 counter.v

reads your netlist

-**v2001** (boolean) force Verilog 2001 mode
-**v1995** (boolean) force Verilog 1995 mode
-**sv** :- read system verilog files

-**vhdl** By default RC reads VHDL - 1993

8.**synthesize -to_mapped -eff low**

-**to_mapped** :- optimizes MUX and datapath and stops before mapping (DEFAULT)
-**to_generic** :- maps specific designs to cells
-**to_incremental-no_incremental** :- turns incremental synthesis on/off

- effort levels are low,medium(default),high

constraints

constraints are the command lines given by the programmer for optimising the system to meet the desired area, power, timing issues.

- Display failed constraints

  echo $::dc ::sdc_failed_commands > failed.sdc

  in the terminal

```
# command failed at line '7' of file 'counter.sdc':
set_atrribute lp_insert_operand_isolation true /
# command failed at line '11' of file 'counter.sdc':
set_attr lp_multi_vt optimization_effort high /
# command failed at line '14' of file 'counter.sdc':
external_delay -input 2  -clk [all_inputs]
# command failed at line '15' of file 'counter.sdc':
external_delay -output 2 -clk [all_outputs]
```

- tightening constraints

path_adjust -delay -200 -from [all::all_seqs] -name pa_i2o
path_adjust -delay -200 -from [all::all_seqs] -name pa_i2c
path_adjust -delay -200 -from [all::all_seqs] -name pa_c2o
path_adjust -delay -200 -from [all::all_seqs] -name pa_c2c

i :- input
o:- output
c :- register

- Remove constraints
  rm [find /des* -exceptions pa_*]
  pa stands for Path Adjust

- Optimising total negative slack (TNS)
  set_attr tns_opto true /

- Sequential optimisation
  set_attr hdl_preserve_unused_register true /
  set_attr delete unloaded_seqs true /

- Sequential merging
  set_attr optimize_merge_flops true /
  set_attr optimize_merge_latches true /
  sequential merging combines equivalent sequential cells, both flops and latches, within same hierarchy. flops increase the area and instance count.

- Mapping to complex cells
  set_attr hdl_ff_keep_feedback true [find / -hdl_arch DFF*]
  uses the complex flip flops inside the library and maps synchronous feedback logic infront of the sequential elements.

- Mixing up the cells into a multibit cell interfacing
  set_attr use_multibit_cells true /
  it makes the circuit more compact, because the dispersed cells are combined into a block using this

command and the multibit cells contain **less power due to less capacitive loading.**

- Removing the undriven pins in a module
  set_attr prune_unused_logic true /
  set_attr prune_unused_logic <path to pins>

- Recombining the logic cuts happened between the combinational circuit and DFF
  set_attr time_recovery_arcs true /
  area optimisations take place when the recovery archs which are unmatched between a flop and a combo.

- Preserve RC mapping on a instance
  set_dont_touch [find /des* -instance alu2]

- Preserving instances and subdesigns
  set_attr preserve true [find /des* -instance alu2]

- Grouping and ungrouping the instances in a system
  group -group_name CRITICAL_GROUP [find / -instance I1] [find / -instance I2]
  ungroup [find / -CRITICAL_GROUP]
  ungroup -threshold 500

- For avoiding the headache of custom partitioning use this , its automatic !
  set_attr auto_partion true /

- Creating a hard region
  set_attribute hard_region true / [find / -instance <INST_NAME>]
  note:- hard regions must be specifed before "_mapped structuring "

- Total deriving environment
  derive_environment <instance> -name <extracted_design_name>
  individual environment of the instance is applied to its actual environment present in the system , its all regarding the slack timing.

- Superthreading
  set_attribute super_thread_servers {machine_names} /
  for automatic
  set_attribute auto_super_thread true /
  by this we can enable different threads to start operating at same time.

- ***set_clock -period 4.75 clock***: clock period constraint set at 4.75 (210 MHz).

- ***set_clock_uncertainty –setup 0.475 clock***: -ve clock skew can lead to setup violations. Possible value of –ve skew is provided to DC so that it can model for that. Generally setup uncertainty is taken as 10%

- ***set_clock_uncertainty –hold 0.27 clock***: +ve clock skew can lead to hold violations. Possible value of +ve skew is provided to DC so that it can model for that. Generally hold uncertainty is taken as 5%.

- ***set_clock_latency 0.45 clock***: this provides possible network latency constraint to DC.

- ***set_clock_latency –source 0.4 clock***: source latency of 0.45 is selected.

- ***set_clock_transition 0.04 clock***: clock transition time of 0.04 is modeled.

- ***set_input_delay 0.40 [all_inputs]***: input delay of 0.4 is set to all inputs.

- ***remove_input_delay [get_ports clock]***: constraining clock with input delay leads to wrong timing analysis. To exclude clock port from the input delay this command is used.

- ***set_output_delay 0.40 [all_outputs]***: output delay of 0.4 is provided. Since all outputs are registered this delay does not affect the timing analysis.

```
##############################################################
############################
# External Delay Information
##############################################################
############################
set_input_delay 0.4 [get_ports {{a_row0[3]}}]
set_input_delay 0.4 [get_ports {{a_row0[2]}}]
set_input_delay 0.4 [get_ports {{a_row0[1]}}]
set_input_delay 0.4 [get_ports {{a_row0[0]}}]
set_input_delay 0.4 [get_ports {{a_row1[3]}}]
```

At the time of compilation

1.reading the TCL file

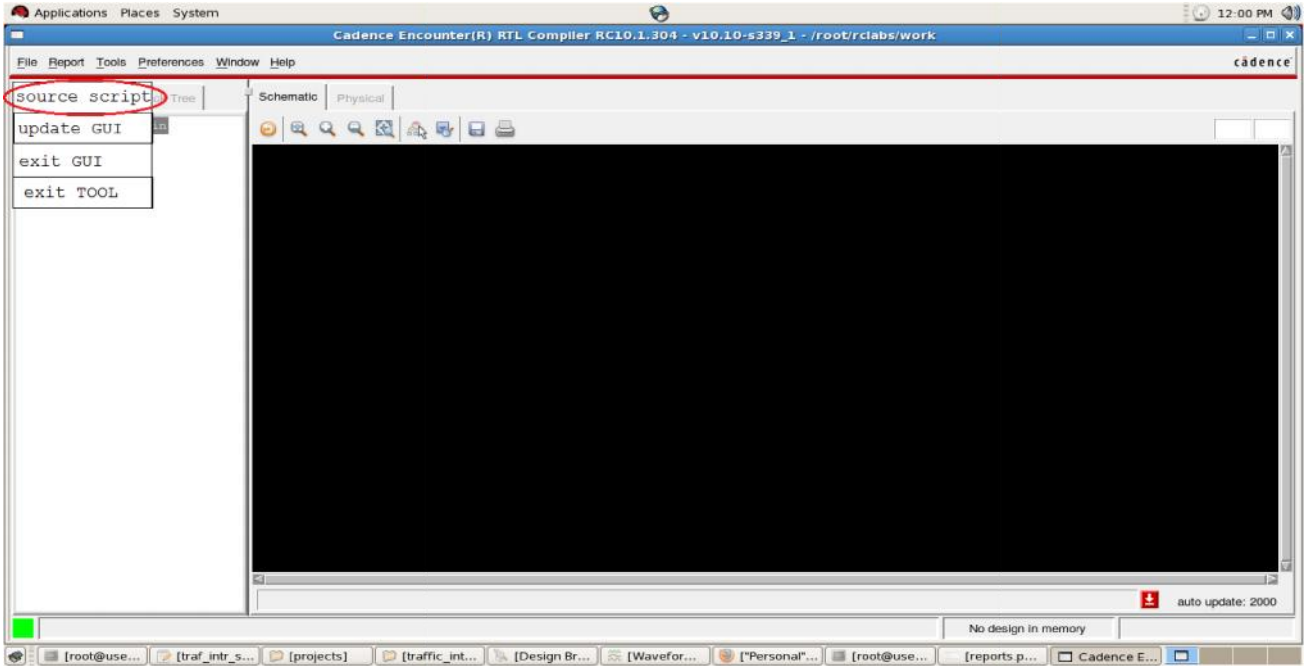


Fig3.1 GUI window

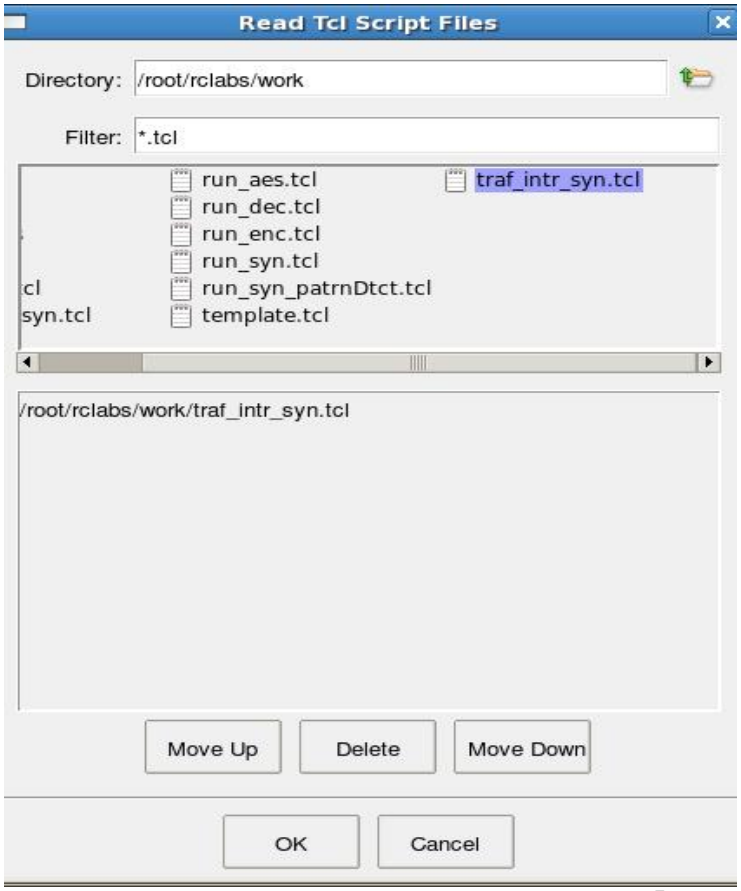


Fig 3.2 Reading "trans_syn.tcl" file

2. reading the report files

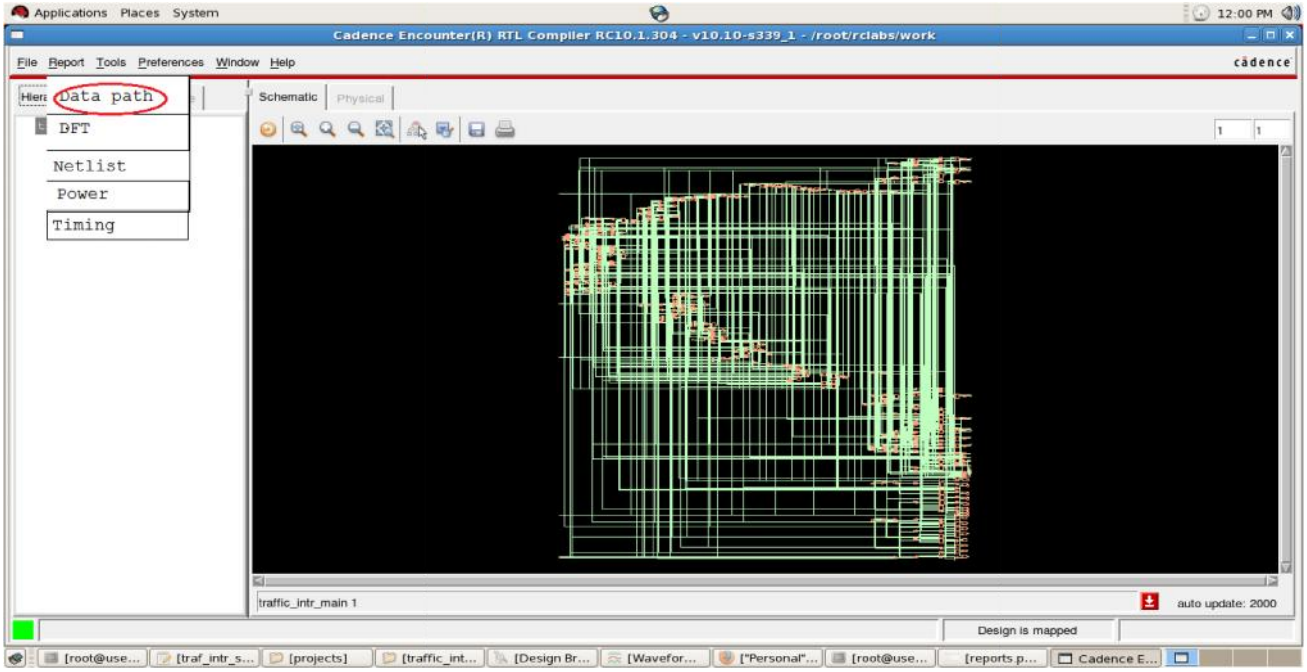


Fig3.3 GUI window

Type	Cell Area	Area %
datapath	0.00	0.00
external	0.00	0.00
others	3174.14	100.00
TOTAL	3174.14	100.00

Fig 3.4 Reading the Report Data path

3. opening the object browser

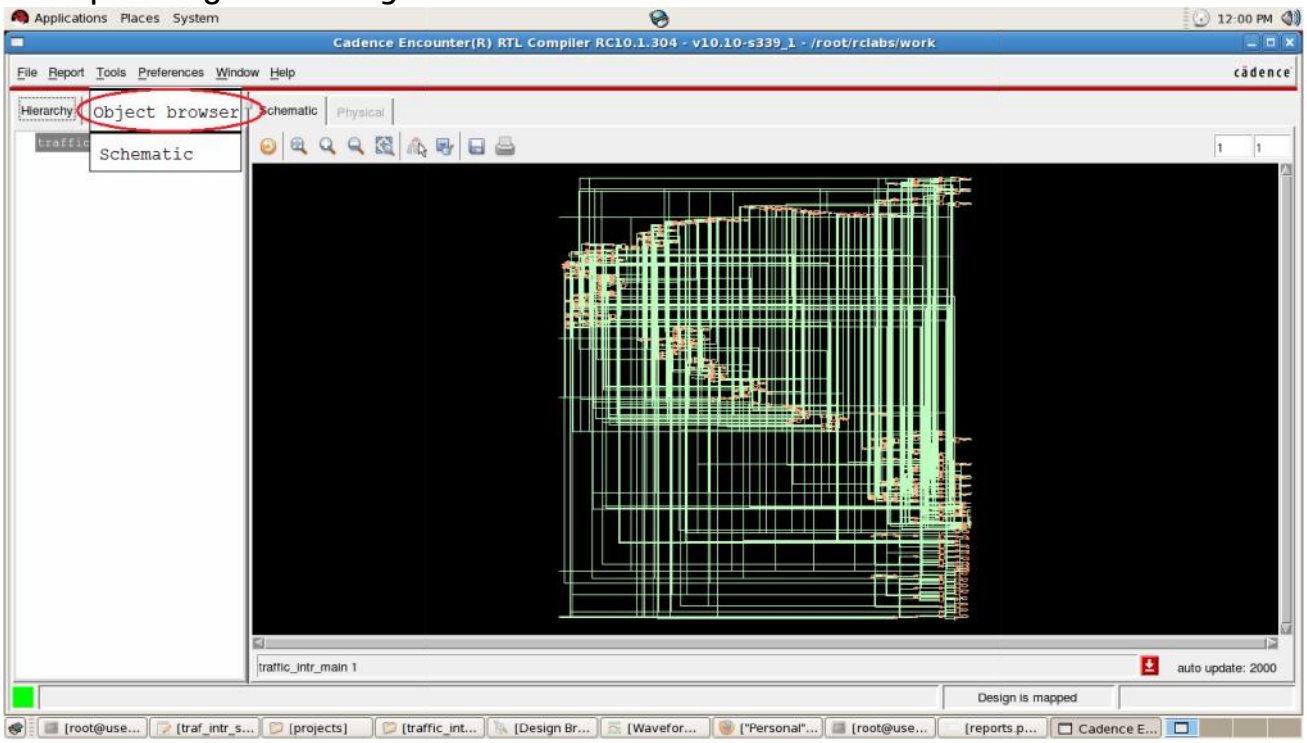


Fig3.5 GUI window

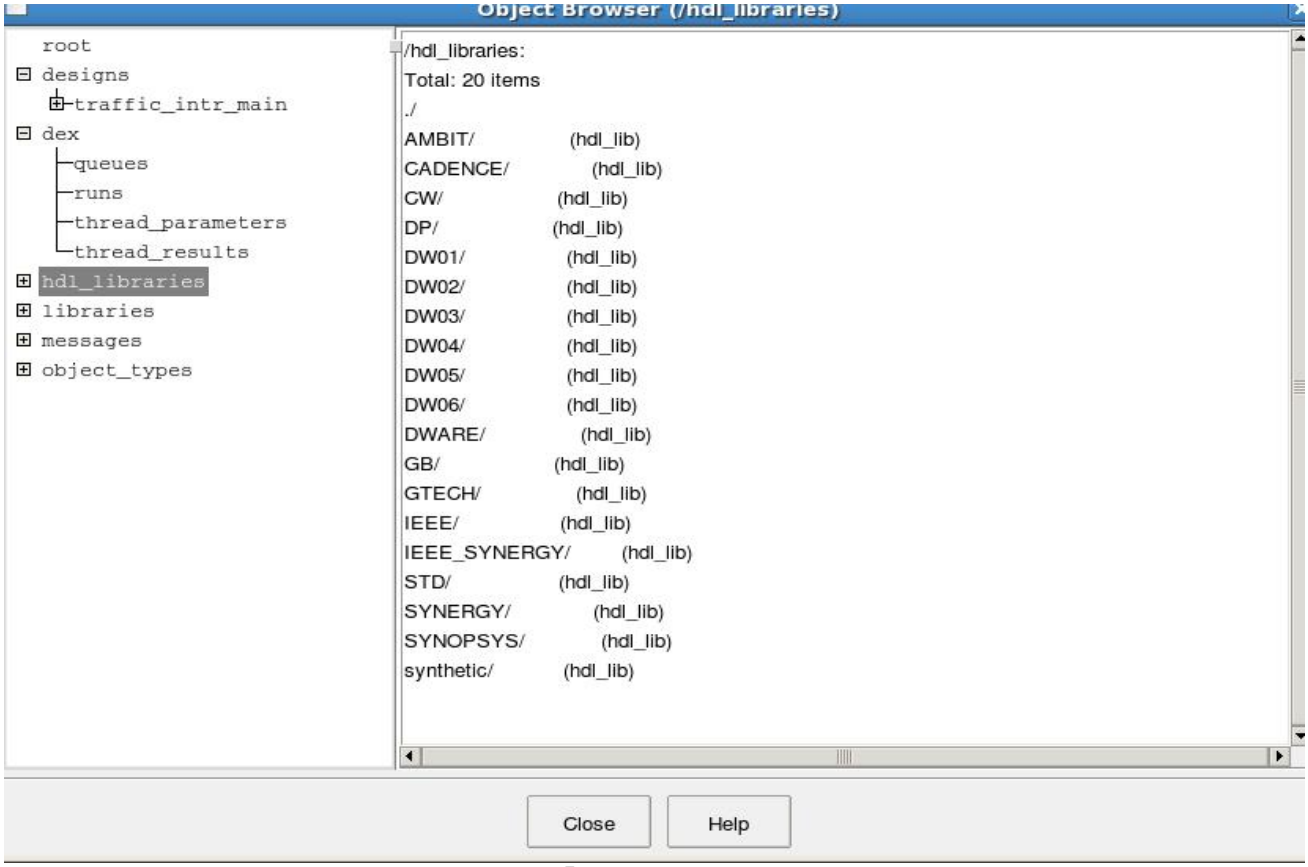


Fig 3.6 Libraries list inside Object Browser

1.**write_template -outfile template.tcl**

Template file will be found in present working directory)

2. **source ./<script_file_name>**

runs the script inside rc(RTL Compiler) , generally used for running templates.

3. **write_hdl > filename.vg**

gate level netlist gets generated over the present directory

4.**write_script > constraints.g**

RC constraints file gets created

5.write_sdc > /root/rclabs/work/ constraints.sdc

all the constraints gets created over a file

Chapter 4

RC OF A 128 BIT COUNTER

An example verilog file: -

```
#########################################################
##### verilog code on  128 bit counter
#########################################################

    module counter(up,down,clk,reset,count);

    input up,down,clk,reset;
    output [127:0]count;
    reg [127:0]count;

    always @(posedge clk or negedge reset)
    begin

     if(!reset)
     count <= 128'b0000;

     else if(up==down)
     count <= count;

     else if(up)
     count <= count+1;

     else if(down)
     count <= count-1;

     else
     count <= count;
    end

endmodule
```

create a tcl file in this way

```
#############################################################
 TCL ( TERMINAL COMMAND LANGUAGE )file counter.tcl
#############################################################

 #read path and library

set_attribute lib_search_path
/root/rclabs/libraries/TIMING/STDCELL (To follow library
path)
set_attribute hdl_search_path /root/rclabs/work
(To follow your Hardware Description Language path)
set_attribute library slow.lib
(To specify the Wanted library eg:- slow,fast,nominal.lib)
#set_attribute information_level 7


# read HDL

read_hdl -v2001 counter.v
(To read your HDL verilog code)


# elaborate design

elaborate


# constraints

define_clock -period 2000 -name clk [find /des* -port clk]
read_sdc counter.sdc
(GIVE your constraints file)


# synthesize

synthesize -to_mapped -eff low
(SPECIFY your effort level as high or low or medium)



# check results
```

```
report power > /root/rclabs/work/reports/counter_power.rpt
report timing >
/root/rclabs/work/reports/counter_timing.rpt
report area > /root/rclabs/work/reports/counter_area.rpt
report design_rules >
/root/rclabs/work/reports/counter_rules.rpt
report gates > /root/rclabs/work/reports/counter_gates.rpt
report datapath >
/root/rclabs/work/reports/counter_datapath.rpt
report clock_gating >
/root/rclabs/work/reports/counter_clkgtng.rpt
report gate -power >
/root/rclabs/work/reports/counter_pwrgt.rpt
report hierarchy >
/root/rclabs/work/reports/counter_hierarchy.rpt
report memory >
/root/rclabs/work/reports/counter_memory.rpt
report instance >
/root/rclabs/work/reports/counter_instance.rpt
report messages >
/root/rclabs/work/reports/counter_messages.rpt
report summary >
/root/rclabs/work/reports/counter_summary.rpt
report qor > /root/rclabs/work/reports/counter_qor.rpt

# generate outputs
echo $::dc::sdc_failed_commands > failed.sdc
write_hdl -m > mjti.vg
(Netlist files are GENERATED as mjti.vg)
write_sdc > mjtinew.sdc
```

THIS IS YOUR CONSTRAINTS FILE

```
#############################################################
CONSTRAINTS FILE (counter.sdc) SYNTHESIS DESIGN
CONSTRAINTS
#############################################################

create_clock -period 2 -name clk [get_ports clk]
set_clock_uncertainty -setup 0.08 [get_clock clk]
set_clock_uncertainty -hold 0.07 [get_clock clk]

set_clock_latency 0.08 -late [get_clocks clk]
set_clock_latency 0.038 -rise [get_clocks clk]

set_max_area 100
set_attribute dp_area_mode true /
set_attribute dp_postmap_downsize true /

set_attribute lp_insert_operand_isolation true /
set_attr lp_insert_clock_gating true /
set_attr max_dynamic_power 100000 /des*/*
set_attr lp_multi_vt_optimisation_effort high /
clock_gating insert_in_netlist

set_attr clock_source_late_latency 2000 [get_clock clk]
set_attr clock_network_late_latency 2000 [get_clock clk ]

set_attr max_fanout 10 [all_inputs]
```

RESULTS

```
##############   failed constraint file    ###########

# command failed at line '7' of file 'counter.sdc':
set_atrribute lp_insert_operand_isolation true /
# command failed at line '11' of file 'counter.sdc':
set_attr lp_multi_vt optimization_effort high /
# command failed at line '14' of file 'counter.sdc':
external_delay -input 500 -clk [all_inputs]
# command failed at line '15' of file 'counter.sdc':
external_delay -output 500 -clk [all_outputs]
```

```
#############################################################
#### REPORT FILES
#############################################################
```

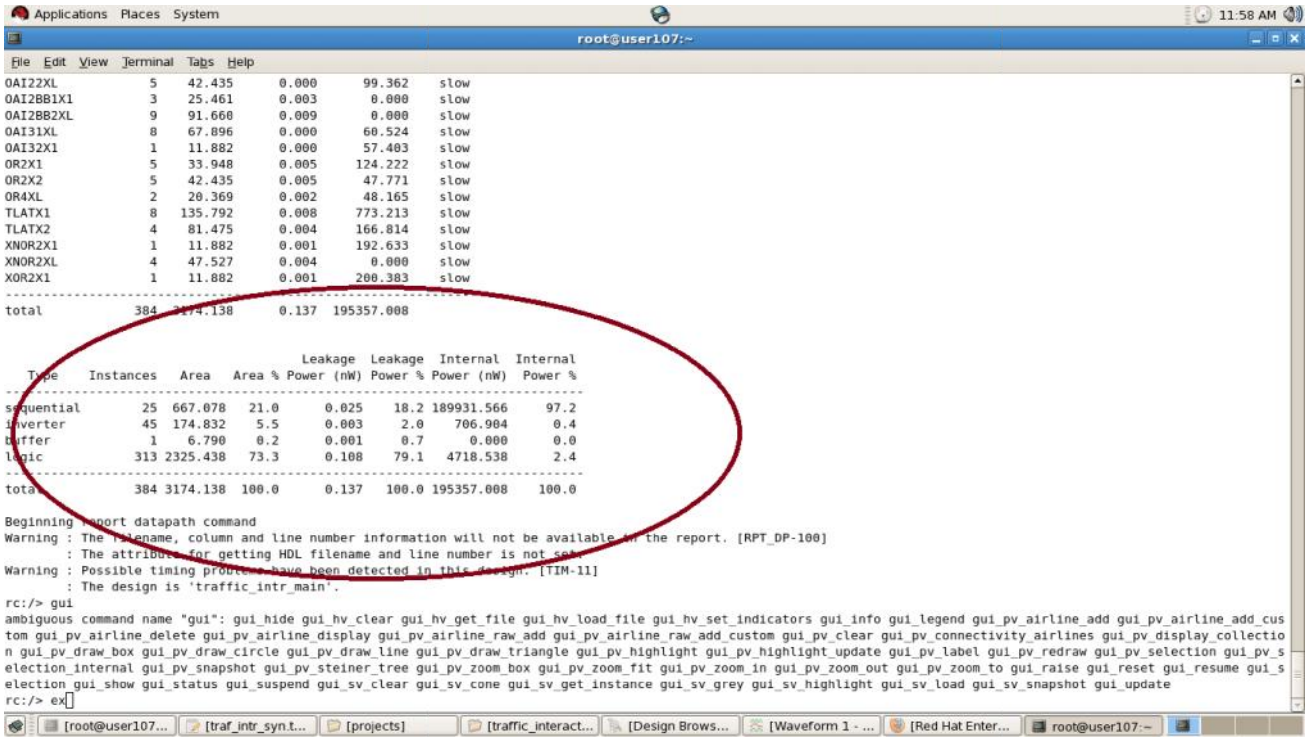


Fig 4.1 Report inside the LOGfile during the process

1.datapath report

```
Command: report datapath >
/root/rclabs/work/report_mj/counter_datapath.rpt
=============================================================
==
  Generated by:           Encounter(R) RTL Compiler
RC10.1.304 - v10.10-s339_1
  Generated on:           Jul 29 2013  06:46:45 pm
  Module:                 mjti
  Technology library:     slow 1.5
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          segmented
  Area mode:              timing library
=============================================================
==


      Type       CellArea Percentage
-------------------------------------
```

```
datapath modules     0.00       0.00
external muxes       0.00       0.00
others            1420.72     100.00
-------------------------------------
total             1420.72     100.00
```

2.design rules report

```
================================================================
==
  Generated by:           Encounter(R) RTL Compiler
RC10.1.304 - v10.10-s339_1
  Generated on:           Aug 03 2013  02:22:34 pm
  Module:                 counter
  Technology library:     slow 1.5
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          segmented
  Area mode:              timing library
================================================================
==

Max_transition design rule: no violations.


Max_capacitance design rule: no violations.


Max_fanout design rule (violation total = 246.000)
Pin                            Fanout           Max
Violation
---------------------------------------------------------------
-----
down (Primary Input)         128.000         5.000
123.000
up (Primary Input)           128.000         5.000
123.000
```

Chapter 5

LIST OF VLSI TOOLS

############## **LIST OF CADENCE TOOLS** ##################

3

* 3D Design Viewer

A

* Allegro AMS Simulator
* Allegro Design Authoring
* Allegro Design Entry Capture / Capture CIS
* Allegro Design Publisher
* Allegro Design Workbench
* Allegro FPGA System Planner
* Allegro Package Designer
* Allegro Package SI
* Allegro PCB Designer
* Allegro PCB Librarian
* Allegro PCB SI
* Allegro System Architect
* AMS Methodology Kit
* Assura Physical Verification

C

* Cadence 3D Design Viewer
* Cadence ActiveParts Portal
* Cadence AMS Methodology Kit
* Cadence Chip Optimizer
* Cadence Chip Planning System
* Cadence CMP Predictor
* Cadence Incisive Verification Kit
* Cadence InCyte Chip Estimator
* Cadence Litho Electrical Analyzer
* Cadence Litho Physical Analyzer
* Cadence Low-Power Methodology Kit
* Cadence MaskCompose Reticle and Wafer Synthesis Suite
* Cadence OrCAD Capture / Capture CIS

* Cadence OrCAD FPGA System Planner
* Cadence OrCAD PCB Designer
* Cadence OrCAD Signal Explorer
* Cadence Palladium Dynamic Power Analysis
* Cadence Palladium series
* Cadence Palladium XP Verification Computing Platform
* Cadence Physical Verification System
* Cadence Process and Proximity Compensation
* Cadence PSpice A/D and Advanced Analysis
* Cadence QRC Extraction
* Cadence QuickView Layout and Manufacturing Data Viewer
* Cadence RF Design Methodology Kit
* Cadence RF SiP Methodology Kit
* Cadence SimVision Debug
* Cadence SiP Co-Design
* Cadence SiP Digital Architect
* Cadence SiP Digital Layout
* Cadence SiP Digital SI
* Cadence SiP Layout
* Cadence Space-Based Router
* Cadence SpeedBridge Adapters
* Cadence Tempus Timing Signoff Solution
* Cadence VIP Catalog
* Chip Optimizer
* Chip Planning System
* CMP Predictor
* C-to-Silicon Compiler

D

* Design IP

E

* Encounter Conformal Constraint Designer
* Encounter Conformal ECO Designer
* Encounter Conformal Equivalence Checker
* Encounter Conformal Low Power
* Encounter DFT Architect
* Encounter Diagnostics
* Encounter Digital Implementation System

* Encounter Library Characterizer
* Encounter Power System
* Encounter RTL Compiler
* Encounter RTL Compiler Advanced Physical Option
* Encounter Timing System
* Encounter True-Time ATPG

F

* First Encounter Design Exploration and Prototyping

I

* Incisive Debug Analyzer
* Incisive Design Team Manager
* Incisive Design Team Simulator
* Incisive Desktop Manager
* Incisive Enterprise Manager
* Incisive Enterprise Simulator
* Incisive Enterprise Specman Elite Testbench
* Incisive Enterprise Verifier
* Incisive Formal Verifier
* Incisive Plan-to-Closure Methodology
* Incisive Software Extensions
* Incisive Verification Kit
* Incisive Xtreme series
* InCyte Chip Estimator

L

* Litho Electrical Analyzer
* Litho Physical Analyzer
* Low-Power Methodology Kit

M

* MaskCompose Reticle and Wafer Synthesis Suite

N

* NanoRoute Router

O

* Open Verification Methodology
* OrCAD Capture and Capture CIS
* OrCAD PCB Designer
* OrCAD Signal Explorer

P

* Physical Verification System
* PSpice A/D and Advanced Analysis

Q

* QRC Extraction
* QuickCycles Service
* QuickView Layout and Manufacturing Data Viewer

R

* Rapid Prototyping Platform
* RF Design Methodology Kit
* RF SiP Methodology Kit

S

* Sigrity BroadBand Spice
* Sigrity OptimizePI
* Sigrity OrbitIO
* Sigrity PowerDC
* Sigrity PowerSI
* Sigrity Speed2000
* Sigrity SystemSI
* Sigrity Transistor-to-behavioral Model Conversion (T2B)
* Sigrity Unified Package Designer (UPD)

* Sigrity XcitePI
* Sigrity XtractIM
* SiP Digital Architect
* SiP Digital Layout
* SiP Digital SI
* SiP Layout
* SoC Encounter RTL-to-GDSII System
* Space-Based Router
* SpeedBridge Adapters

V

* Virtual System Platform
* Virtual System Platform for the Xilinx Zynq-7000 EPP
* Virtuoso Accelerated Parallel Simulator
* Virtuoso AMS Designer
* Virtuoso Analog Design Environment
* Virtuoso Chip Assembly Router
* Virtuoso DFM
* Virtuoso Digital Implementation
* Virtuoso Layout Migrate
* Virtuoso Layout Suite
* Virtuoso Layout Suite for Electrically Aware Design
* Virtuoso Liberate
* Virtuoso Liberate LV
* Virtuoso Liberate MX
* Virtuoso Multi-Mode Simulation
* Virtuoso Passive Component Designer
* Virtuoso Power System
* Virtuoso RF Designer
* Virtuoso Schematic Editor
* Virtuoso SiP Architect
* Virtuoso Spectre Circuit Simulator
* Virtuoso UltraSim Full-Chip Simulator
* Virtuoso Variety
* Virtuoso Visualization and Analysis
* VoltageStorm Power Verification

courtesy : Cadence official website

1.VLSI tools (chip design, simulation, and layout)

* **icms** - command to start an IC design desktop. This tool starts all the others listed below.

* **Composer** - schematic capture

* **Analog Artist** - Analog/Mixed Signal simulation framework and GUI. It is similar, but far more powerful, to PSpice Probe.

* **cdsSPICE** - analog SPICE simulator.

* **Spectre** - (analysis on DC , AC ....) analog simulator, not based on the original Berkeley SPICE. It is still a Newton-Raphson engine, however. It does behavioral models as well as standard MOSFET, BJT, and others. I use it in preference to cdsSPICE, Berkeley SPICE, or PSpice.

* **Virtuoso** - layout editor

* **DFII, Design Framework II** - Database system that Cadence uses for all IC tools. Basically, all designs can be hierarchical have several views. Views can be schematics, layouts, behavioral descriptions, etc.

2. VHDL (hardware description language) tools:

* **hdldesk** - GUI desktop for HDL tools. This calls Leapfrog, cv, and ev if you want it to.
* **Leapfrog** - simulator
* **cv** - compiler
* **ev** - elaborator (VHDL term for linker)

3. Systems Tools (circuit board level stuff)

 * **brddesign** - GUI desktop for system tools.
 * **Concept** - schematic capture
 * **Allegro** - PCB layout
 * **Analog Workbench** - simulator interface with virtual oscilloscopes, spectrum analyzers, etc. Uses Concept for schematic entry.

courtesy :UofA Cadence

##

TRENDING SYNTHESIS RTL COMPILERS IN THE PRESENT MARKET

##

Software tools for logic synthesis targeting ASICs

* Design Compiler by Synopsys
* Encounter RTL Compiler by Cadence Design Systems
    o BuildGates, an older product by Cadence Design Systems, humorously named after Bill Gates
* Talus Design by Magma Design Automation
* RealTime Designer by Oasys Design Systems
* BooleDozer: Logic synthesis tool by IBM (internal IBM EDA tool)

Software tools for logic synthesis targeting FPGAs

* XST (delivered within ISE) by Xilinx
* Quartus II integrated Synthesis by Altera
* IspLever by Lattice Semiconductor
* Encounter RTL Compiler by Cadence Design Systems
* LeonardoSpectrum and Precision (RTL / Physical) by Mentor Graphics
* Synplify (PRO / Premier) by Synopsys
* Blast FPGA by Magma Design Automation